Remarks

As stated above, Applicant appreciates the Examiner's thorough examination of the subject application and requests reexamination and reconsideration of the subject application in view of the preceding amendments and the following remarks.

As of the office action of July 25, 2008, claims 4-6, 8-18, 20, 22, and 24-27 are pending in the subject application, of which claims 4, 15-16, 20, 22 and 24-26 are independent claims. With this response applicant has amended claims 4-6, 8-18, 20, 22, and 24-27. Applicant respectfully submits that no new matter is being added as a result of these amendments.

As an initial matter, the Examiner has rejected claim 4 under 35 USC §112 as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Applicant respectfully traverses this rejection.

Specifically, the Examiner states that it "is unclear from Applicant's language what the program fragment consists of, whether the program also consists of the subprogram, and whether the series of instructions is part of the program, subprogram, or both." *Official Action*, page 2. Applicant respectfully disagrees with the Examiner's assertion. Applicant's claim 4 is provided below for the Examiner's convenience.

> 7. (Currently Amended) A method of verifying a program fragment downloaded onto a reprogrammable embedded system, equipped with a rewritable memory, a microprocessor and a virtual machine equipped with an execution stack and with operand registers, said program fragment consisting of an object code and including at least one subprogram consisting of a series of instructions manipulating said operand registers, said microprocessor and virtual machine configured to interpret said object code, said embedded system being interconnected to a reader, wherein subsequent to a detection of a downloading command and a storage of said object code in said rewritable memory, said method, for each subprogram, comprises:
>
> initializing a type stack and a table of register types through data representing a state of said virtual machine on initialization of an execution of said temporarily stored object code;

> carrying out a verification process of said temporarily stored object code instruction by instruction, by discerning an existence, for each current instruction, of a target, a branching-instruction target, a target of an exception-handler call or a target of a subroutine call, and, said current instruction being a target of a branching instruction, said verification process including verifying that said stack is empty and rejecting said program fragment otherwise;
>
> verifying and updating an effect of said current instruction on said data types of said type stack and of said table of register types;
>
> said verification process being successful when said table of register types is not modified in the course of a verification of all the instructions, and said verification process being carried out instruction by instruction until said table of register types is stable, with no modification being present, said verification process being interrupted and said program fragment being rejected, otherwise. Applicant's *claim 4*, *Emphasis Added.*

As shown in claim 4 above, the program fragment consists of an object code and includes at least one subprogram consisting of a series of instructions manipulating the operand registers. In other words, the subprogram consists of a series of instructions and is part of the program fragment. As such and in light of the above, Applicant respectfully submits that no further correction is required and requests that the rejection under 35 USC §112 be withdrawn.

Claims 4-6, 8-18, 20, 22, and 24-27 stand rejected under 35 U.S.C. 103(a) as being unpatentable over Gosling (US 6,075,940) in view of Scandura (US 6,275,976) Applicant respectfully traverses this rejection.

Gosling appears to describe a verification process for Java code. Specifically, Gosling appears to teach implementing a method of verifying the integrity of bytecode language written programs in which "a virtual operand stack for temporarily storing stack information indicative of data stored in a program operand stack during the execution of a specified bytecode program." *See Gosling*, col. 2, lines 11-14. It is Applicant's understanding that the verifier as disclosed by Gosling is said to process the specified program by sequentially processing each byte code

instruction, updating the virtual operand stack to indicate the number, sequence and data types of data that would be stored in the operand stack at each point in the program. *See Gosling*, col. 2, lines 15-19. Further, a comparison of the virtual stack information with data type restrictions of each byte code instruction to check data inconsistent with the data type restrictions and possible underflow or overflow of the operand stack generated by byte codes instruction is also executed. *See Gosling*, col. 2, lines 19-26. In other words, Gosling appears to teach the implementation of a virtual operand stack, which is added and compared with the data type restrictions.

In contrast, claim 4 of the subject application states "said verification process including verifying that said stack is empty and rejecting said program fragment otherwise" and "verifying and updating an effect of said current instruction on said data types of said type stack and of said table of register types." Here, the verification process is carried out instruction by instruction and is deemed successful in the absence of modification when the table of the register types is not modified, otherwise the applet is rejected and an interruption occurs.

Although Gosling appears to teach discriminating multiple entry-points, the verifier appears to operate by taking a "snapshot" of the operand stack prior to each multiple-entry point and compares this "snapshot" with the virtual operand stack states, after having processed each of the preceding byte code instructions for the current multiple-entry point, a program fault being generated if the virtual stack states are not identical. *See Gosling*, col. 2, lines 27-42.

Moreover, Gosling appears to teach a verification process for bytecode (i.e., object code). As the Examiner states on page 2 of the Official Action, Gosling does not teach that the verification was successful only if the table of register types is unmodified during the single pass over all the instructions. Furthermore, Gosling does not appear to teach verifying that the stack is empty whenever the current instruction is a branching instruction, instead verifying branching

targets using a snapshot of the virtual stack state at the branching point. *See Gosling*, col. 19, lines 25-45. Applicant respectfully submits that this is an essential new and inventive feature of the claimed invention as it allows the verification to be performed by an embedded system with little memory and processing power. This issue does not appear to have been addressed by Gosling and is not taught nor even suggested by the Scandura reference.

Applicant respectfully submits that the Scandura reference teaches consistency verifications (col. 34, lines 19-56) that apply between program specifications and program source, not to object code. Thus, Applicant respectfully submits that Scandura's consistency verification is not relevant in the context of Applicant's newly amended claims.

Moreover, Scandura appears to concern higher order software. *See Scandura*, col. 14, lines 50-53. Scandura appears to be directed towards providing human designers with automated support for specifying, designing, implementing and maintaining arbitrarily large, complex, software systems that are both internally consistent and logically correct. *See Scandura*, col. 14, lines 6-11.

Such an object may be obtained by implementing specifications and solution designs via simultaneous and successive levels of data and process refinement, capitalizing in semantically meaningful relationships between different kinds of specification and design refinement, partitioning input and output values and types into equivalence clauses. *See Scandura*, col. 14, lines 13-19.

In contrast, the subject application is directed towards verification of object code performed by embedded computers, as well as object code transformations performed by unattended software. The subject application is also specifically dedicated to perform

verification and transformation automatically and on object code alone without recourse to high-level specification.

Therefore, Applicant is unable to see how a person or ordinary skill in the art would adapt the teachings of Scandura, which is designed to help humans create large software, to Gosling's verification method, that could possibly teach or suggest the automatic claimed method of the invention (unattended process, size constraints of embedded systems, absence of high level specifications). As such, Applicant respectfully submits that independent claim 4 is in condition for allowance.

Independent claims 15-16, 20, 22 and 24-26 include similar limitations and are believed to be in condition for allowance as well. Since the remaining dependent claims depend, either directly or indirectly, from Applicant's independent claims, Applicant respectfully submits that these claims are in condition for allowance as well.

Further, Applicant respectfully submits that claim 9 requires the verification to reject program fragments where the target of a subroutine call follows an instruction that may not trigger a transfer of control (unconditional branching, subroutine return or withdrawal of exception). Applicant respectfully submits that Gosling fails to teach or suggest any such verification. *See Gosling*, col. 14, lines 55-67.

In addition, Applicant respectfully submits that claim 10 requires the verification to reject program fragments where an instruction is the target of multiple incompatible branchings. Gosling fails to teach or suggest any such verification. *See Gosling*, col. 14, lines 55-67.

Regarding claims 15-17, these claims concern an object code transformation process to be performed before the object code is transferred to the embedded computer, as opposed to the

verification process on said computer. Thus, Applicant disagrees that the claims "are substantially the same" as indicated by the Examiner.

Having overcome all of the outstanding objections and rejections, Applicant respectfully submits that the application is now in condition for allowance. Early allowance of the application is respectfully requested.

Applicant does not believe that any additional fees are necessitated by this response. However, in the event any additional fees are due, please charge our Deposit Account No. 50-2324 for any necessary fees, referencing Attorney Docket No. 102114.00034.

The Examiner is invited to telephone Applicants' attorney (@617-305-2143) to facilitate prosecution of this application.

Respectfully submitted,

Date: October 27, 2008

/Brian J Colandreo/
Brian J Colandreo (Reg. No. 42,427)

Holland & Knight LLP

10 St. James Avenue
Boston, MA 02116-3889
Telephone: 617-305-2143
Facsimile: 617-523-6850

5740086_v1